NAME : HARSHIT SINGHANIA
ROLL : 2105890
LAB ASSIGNMENT 4

LE

1 Write a menu driven program to perform the following operations in a single linked list by using suitable user
defined functions for each case.

a) Traversal of the list

b) Check if the list is empty

c) Insert a node at the certain position (at beginning/end/any position)

d) Delete a node at the certain position (at beginning/end/any position)

e) Delete a node for the given key

f) Count the total number of nodes

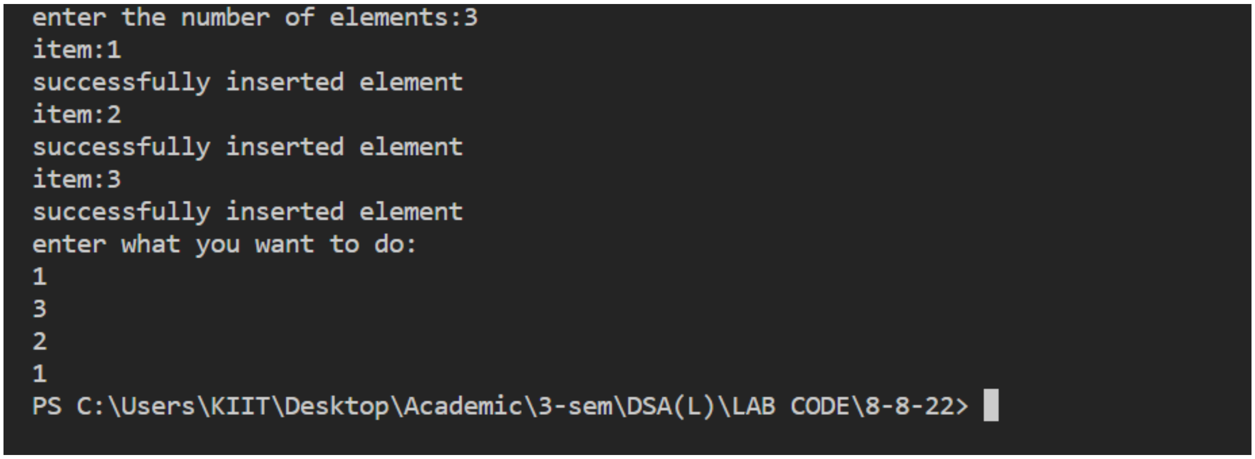
g) Search for an element in the linked list

Verify & validate each function from main method.

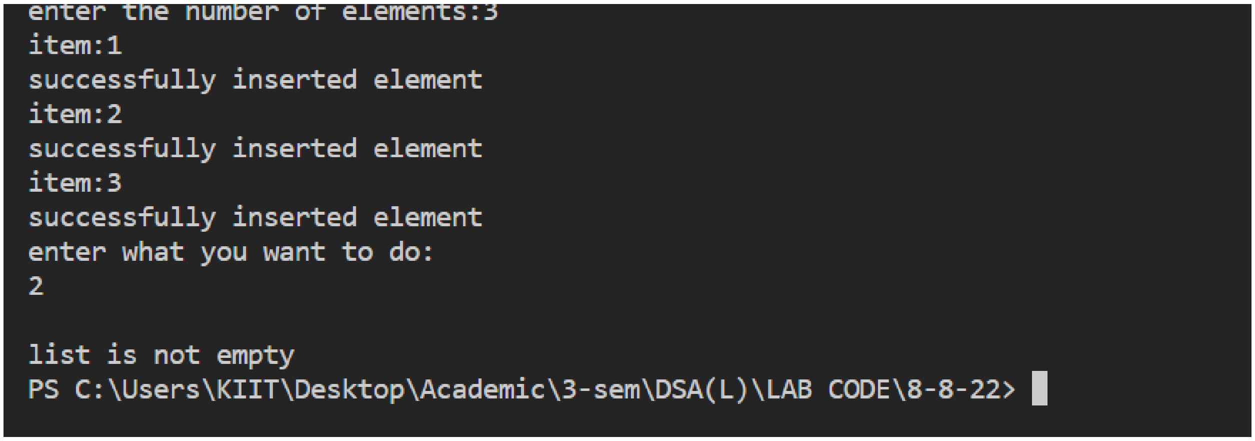
```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int data;
   struct node *next;
};

struct node *head = NULL;

void createNode(int item)
{
   struct node *ptr = (struct node *)malloc(sizeof(struct node *));
   if (ptr == NULL)
   {
      printf("memory not allocated");
      return;
```

```
    }
    else
    {
        ptr->data = item;
        ptr->next = head;
        head = ptr;
        printf("successfully inserted element \n");
    }
}

void traverseList()
{
    struct node *ptr = head;
    while (ptr != NULL)
    {
        printf("%d ", ptr->data);
        ptr = ptr->next;
        printf("\n");
    }
}

void isEmpty()
{
    if (head == NULL)
    {
        printf("\nlist is empty\n");
    }
    else
    {
        printf("\nlist is not empty\n");
    }
}

int getSize(struct node *node)
{
    int size = 0;
    while (node != NULL)
    {
        node = node->next;
        size++;
    }
    return size;
}
```

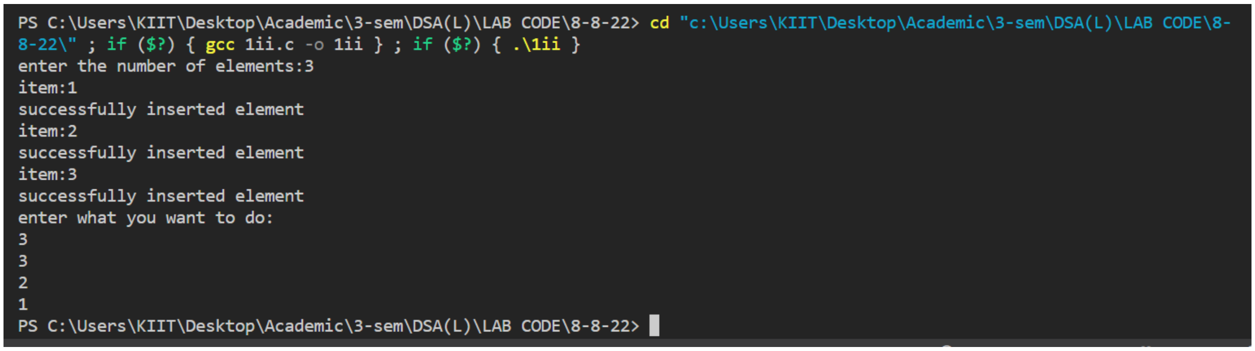
```
void insertNode(int n, int data, struct node **head)
{
    int size = getSize(*head);
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->next = NULL;
    if (n < 0 || n > size)
    {
        printf("\nInvalid position");
        return;
    }
    else if (n == 0)
    {
        newNode->next = *head;
        *head = newNode;
    }
    else
    {
        struct node *temp = *head;
        while (--n)
        {
            temp = temp->next;
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

void deleteNode (struct node** head_ref, int position) {
    if (*head_ref == NULL) {
        printf("\nList is empty");
        return;
    }
    struct node* temp = *head_ref;
    if (position == 0) {
        *head_ref = temp->next;
        free(temp);
        return;
    }
    for (int i=0; temp!=NULL && i<position-1; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("\nInvalid position");
        return;
    }
    struct node* next = temp->next->next;
    free(temp->next);
    temp->next = next;
}

void deleteNodeKey(struct node** head_ref, int position) {
    if (*head_ref == NULL) {
        printf("\nList is empty");
        return;
    }
    struct node* temp = *head_ref;
    if (position == 0) {
        *head_ref = temp->next;
        free(temp);
        return;
    }
    for (int i=0; temp!=NULL && i<position-1; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        printf("\nInvalid position");
        return;
    }
    struct node* next = temp->next->next;
    free(temp->next);
    temp->next = next;
}

void countNodes() {
    struct node *temp = head;
    int count=0;
    while (temp != NULL) {
        temp = temp->next;
        count++;
    }
    printf("\n Number of nodes in the list is %d \n", count);
}

int searchElements(struct node* head, int item, int index) {
    if (head == NULL) {
        return -1;
    }
    if (head->data == item) {
        return index;
    }
    index++;
    return searchElements(head->next, item, index);
}

int main()
{
    int item, n, i;
    printf("enter the number of elements:");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("item:");
        scanf("%d", &item);
        createNode(item);
    }
    int choice;
    printf("enter what you want to do:\n");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            traverseList();
            break;
        case 2:
            isEmpty();
            break;
        case 3:
            insertNode(1, 10, &head);
            break;
        case 4:
            deleteNode(&head, 1);
            break;
        case 5:
            deleteNodeKey(&head, 1);
            break;
        case 6:
            countNodes();
            break;
        case 7:
            int call = searchElements(head, 10, 0);
            printf("\n%d", call);
            break;
        default:
            printf("\nInvalid choice");
            break;

    }
}
```



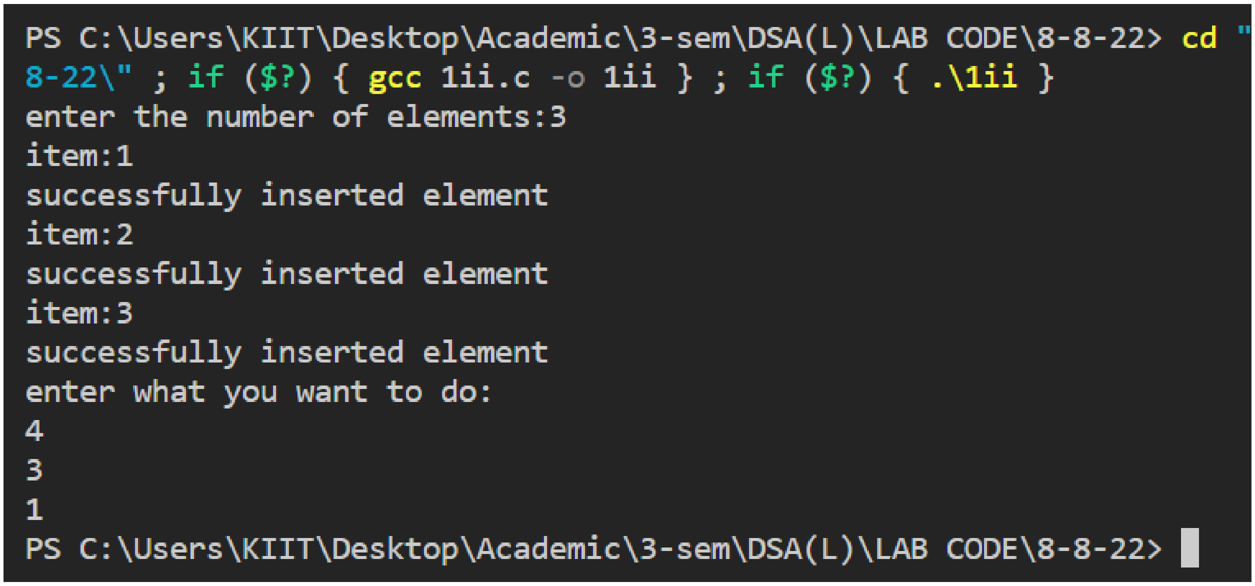
```
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
1
3
2
1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> 
```



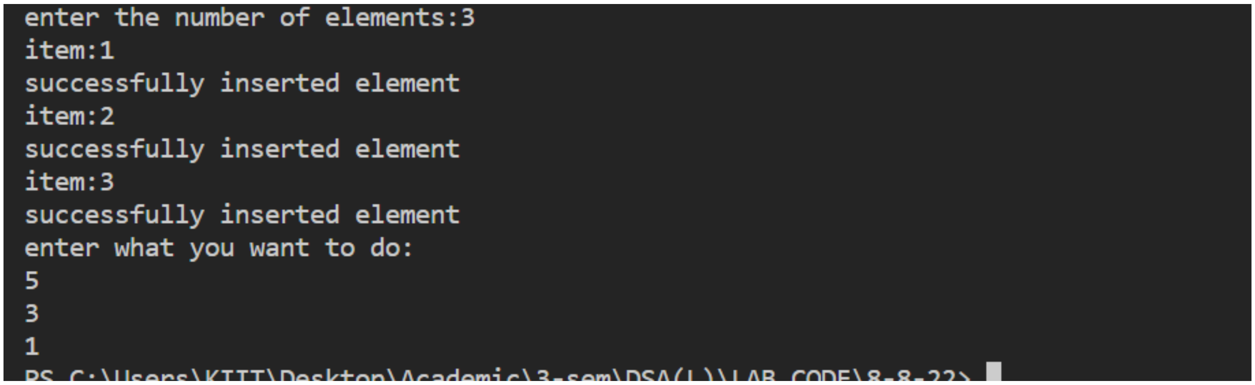
```
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
2

list is not empty
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> 
```



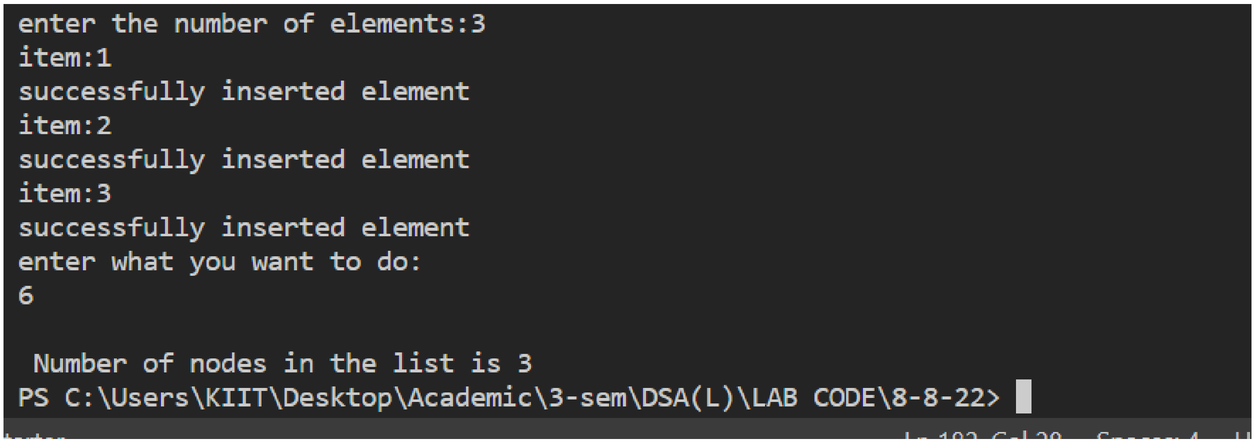
```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "c:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-
8-22\" ; if ($?) { gcc 1ii.c -o 1ii } ; if ($?) { .\1ii }
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
3
3
2
1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> 
```



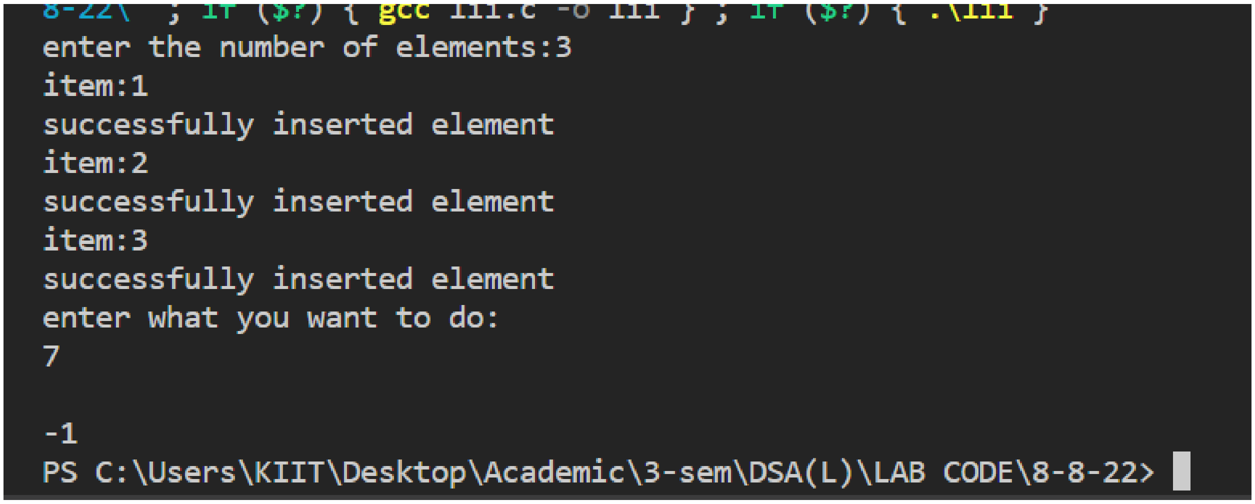
```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "
8-22\" ; if ($?) { gcc 1ii.c -o 1ii } ; if ($?) { .\1ii }
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
4
3
1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```



```
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
5
3
1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```



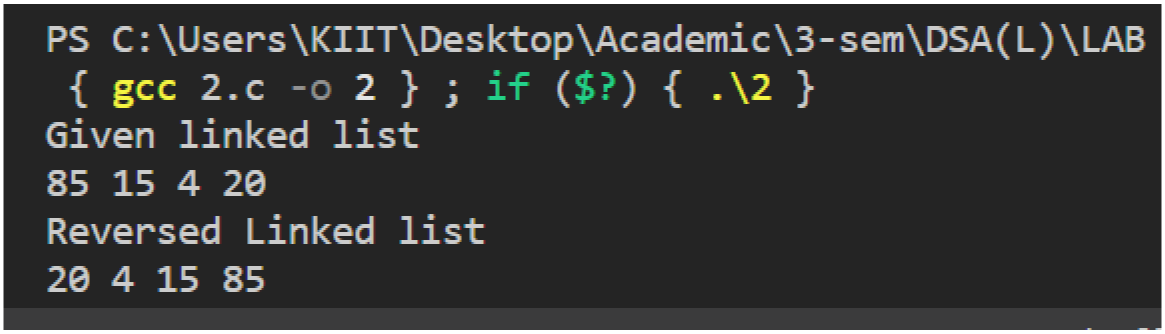
```
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
6

 Number of nodes in the list is 3
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```



```
8-22\  ; if ($?) { gcc 111.c -o 111 } ; if ($?) { .\111 }
enter the number of elements:3
item:1
successfully inserted element
item:2
successfully inserted element
item:3
successfully inserted element
enter what you want to do:
7

-1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

2. WAP to display the contents of a linked list in reverse order.

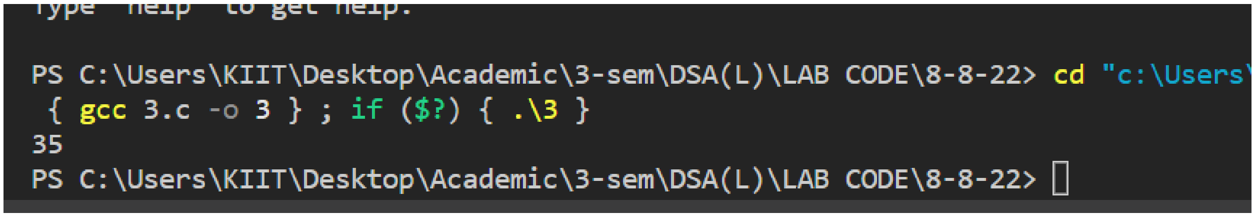
```
// Iterative C program to reverse a linked list
#include <stdio.h>
#include <stdlib.h>

/* Link list node */
struct Node {
	int data;
	struct Node* next;
};

/* Function to reverse the linked list */
static void reverse(struct Node** head_ref)
{
	struct Node* prev = NULL;
	struct Node* current = *head_ref;
	struct Node* next = NULL;
	while (current != NULL) {
		// Store next
		next = current->next;

		// Reverse current node's pointer
		current->next = prev;

		// Move pointers one position ahead.
		prev = current;
		current = next;
	}
	*head_ref = prev;
}

/* Function to push a node */
void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node
        = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

/* Function to print linked list */
void printList(struct Node* head)
{
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

/* Driver code*/
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;

    push(&head, 20);
    push(&head, 4);
    push(&head, 15);
    push(&head, 85);

    printf("Given linked list\n");
    printList(head);
    reverse(&head);
    printf("\nReversed Linked list \n");
    printList(head);
    getchar();
}
```



```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB
 { gcc 2.c -o 2 } ; if ($?) { .\2 }
Given linked list
85 15 4 20
Reversed Linked list
20 4 15 85
```

3 WAP to print mth node from the last of a linked list of n nodes.

```
#include <stdio.h>
#include <stdlib.h>

/* Link list node */
typedef struct Node {
        int data;
        struct Node* next;
}Node;

/* Function to get the nth node from the last of a linked list*/
void printNthFromLast(Node* head, int n)
{
        int len = 0, i;
        Node* temp = head;

        // count the number of nodes in Linked List
        while (temp != NULL) {
                temp = temp->next;
                len++;
        }
        if (len < n)
                return;
        temp = head;
        for (i = 1; i < len - n + 1; i++)
                temp = temp->next;
        printf("%d",temp->data);
        return;
}

void push(struct Node** head_ref, int new_data)
{
        Node* new_node = (Node *)malloc(sizeof(Node));
        new_node->data = new_data;


        new_node->next = (*head_ref);

        (*head_ref) = new_node;
}

// Driver Code
int main()
{

        /* Start with the empty list */
        struct Node* head = NULL;

        // create linked 35->15->4->20
        push(&head, 20);
        push(&head, 4);
        push(&head, 15);
        push(&head, 35);
        printNthFromLast(head, 4);
        return 0;
}
```



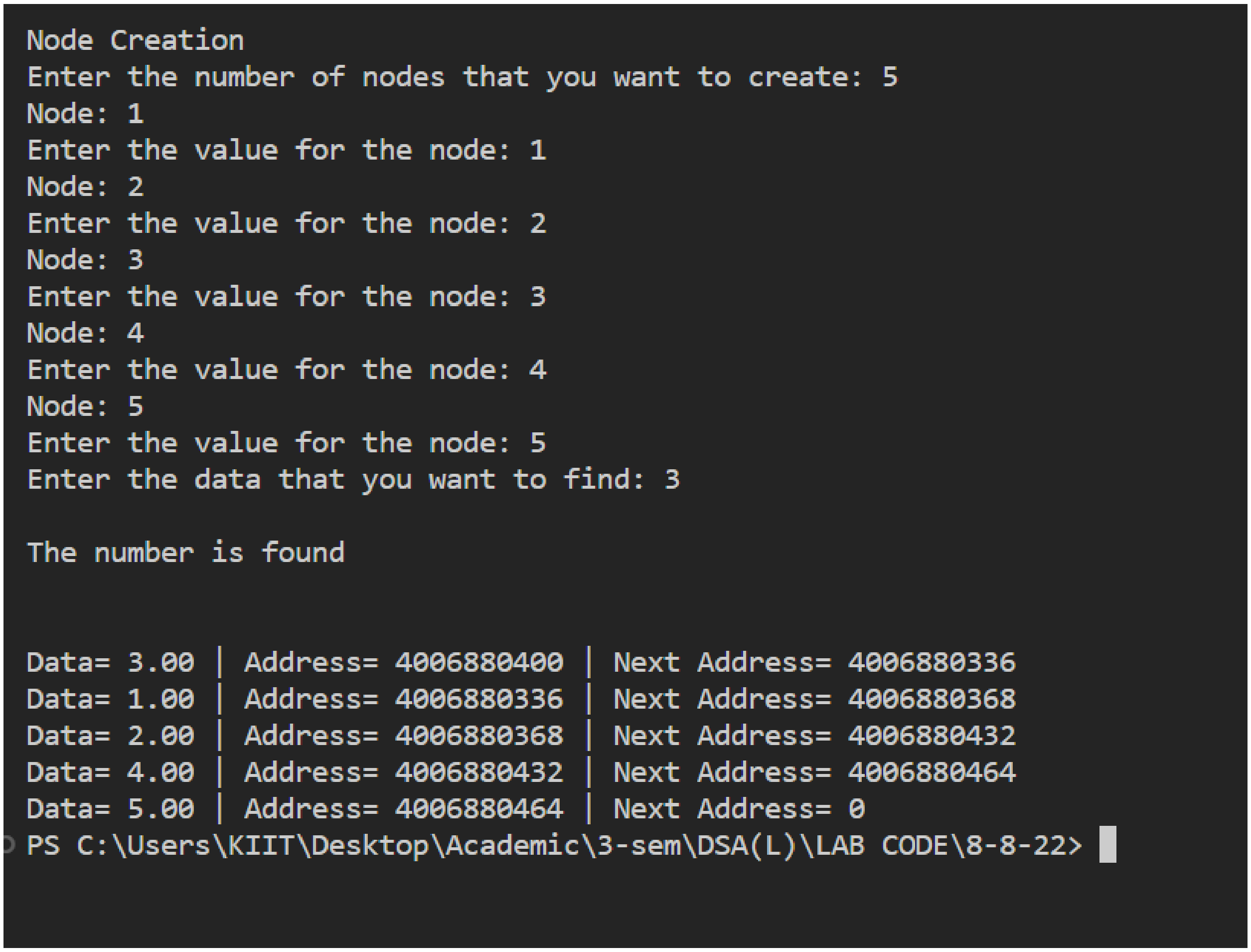
```
Type 'help' to get help.

PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "c:\Users
 { gcc 3.c -o 3 } ; if ($?) { .\3 }
35
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> 
```

HEs

1. WAP to search an element in a simple linked list, if found delete that node and insert that node at beginning. Otherwise display an appropriate Message.

```
#include<stdio.h>
#include<stdlib.h>

struct node{
    double data;
    struct node *next;

};
struct node *head=NULL;
struct node *tail=NULL;

void task1(double x);
void task2(struct node **h, double x);
void task3(struct node **h, int x, double y);
void task4(struct node *h);

int main()
{
    int n, choice, key, i;
    struct node *cur;
    double n1;

    printf("\n\n");
    printf("Node Creation ");
    printf("\n");

    printf("Enter the number of nodes that you want to create: ");
    scanf("%d", &n);

    for (i=0; i<n; i++)
    {
        printf("Node: %d\n", i+1);
        cur=(struct node *)malloc(sizeof(struct node));
        printf("Enter the value for the node: ");
        scanf("%lf", &cur->data);
        cur->next=NULL;
        if (head==NULL){
            tail=head=cur;
        }
        else{
            tail->next=cur;
            tail=cur;
        }
    }

    printf("Enter the data that you want to find: ");
    scanf("%lf", &n1);

    task1(n1);
    return 0;
}

void task1(double x)
{
    struct node *ptr;
    double temp;
    int count=0, pos=0;

    for (ptr=head; ptr!=NULL; ptr=ptr->next)
    {
        if (ptr->data==x)
        {
            printf("\nThe number is found\n");
            temp=x;
            count++;
            break;
        }
    }
    if (count==0)
        printf("\nData not found\n");

    printf("\n\n");
    if (temp>0 || temp<0)
    {
        task2(&head, temp);
        task3(&head, pos, temp);
        task4(head);
    }
}

void task2(struct node **h, double x)
{
    struct node *ptr, *prv;

    if (*h==NULL)
    {
        printf("The list is empty\n");
    }
    else
    {
        ptr=*h;
        while (ptr!=NULL)
        {
            if (ptr->data==x)
                break;
            else
            {
                prv=ptr;
                ptr=ptr->next;
            }
        }
        if (ptr==NULL)
        {
            printf("The data is not found\n");
        }
        else if (ptr==*h)
        {
            *h=ptr->next;
            free(ptr);
        }
        else
        {
            prv->next=ptr->next;
            free(ptr);
        }
    }
}

void task3(struct node **h, int x, double y)
{
    struct node *cur, *ptr;

    cur=(struct node *)malloc(sizeof(struct node));
    cur->data=y;
    cur->next=NULL;

    if (*h==NULL)
    {
        *h=cur;
    }
    else if (x==0)
    {
        cur->next=*h;
        *h=cur;
    }
}
```

```
void task4(struct node *h)
{
   for (h; h!=NULL; h=h->next)
   {
      printf("Data= %.2lf | Address= %u | Next Address= %u\n",
                                          h->data, h, h->next);
   }
}
```

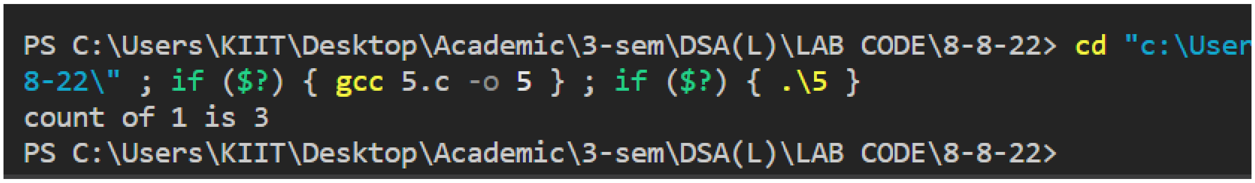


```
Node Creation
Enter the number of nodes that you want to create: 5
Node: 1
Enter the value for the node: 1
Node: 2
Enter the value for the node: 2
Node: 3
Enter the value for the node: 3
Node: 4
Enter the value for the node: 4
Node: 5
Enter the value for the node: 5
Enter the data that you want to find: 3

The number is found


Data= 3.00 | Address= 4006880400 | Next Address= 4006880336
Data= 1.00 | Address= 4006880336 | Next Address= 4006880368
Data= 2.00 | Address= 4006880368 | Next Address= 4006880432
Data= 4.00 | Address= 4006880432 | Next Address= 4006880464
Data= 5.00 | Address= 4006880464 | Next Address= 0
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

2. WAP to count the number of occurrences of an element in a linked list of n nodes.

```
#include <stdio.h>
#include <stdlib.h>

/* Link list node */
struct Node
```

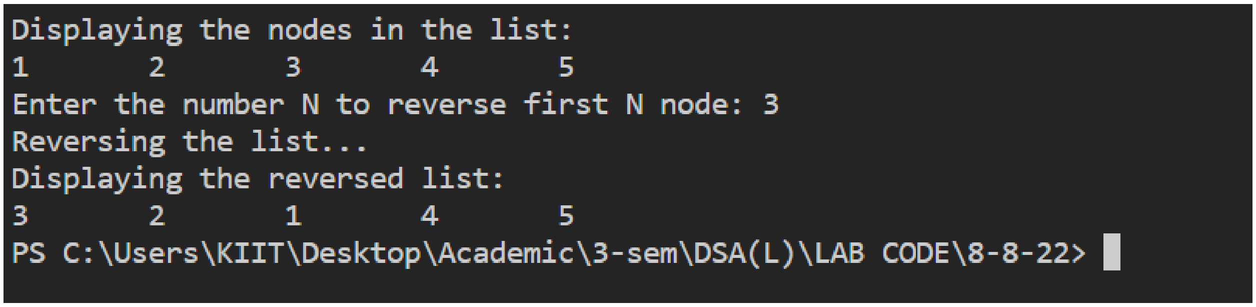
```
{
    int data;
    struct Node *next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front
of the list. */
void push(struct Node **head_ref, int new_data)
{
    /* allocate node */
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Counts the no. of occurrences of a node
(search_for) in a linked list (head)*/
int count(struct Node *head, int search_for)
{
    struct Node *current = head;
    int count = 0;
    while (current != NULL)
    {
        if (current->data == search_for)
            count++;
        current = current->next;
    }
    return count;
}

/* Driver program to test count function*/
int main()
{
    /* Start with the empty list */
    struct Node *head = NULL;
    /* Use push() to construct below list
    1->2->1->3->1 */
    push(&head, 1);
    push(&head, 3);
    push(&head, 1);
    push(&head, 2);
    push(&head, 1);

    /* Check the count function */
    printf("count of 1 is %d", count(head, 1));
    return 0;
}
```



```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "c:\User
8-22\" ; if ($?) { gcc 5.c -o 5 } ; if ($?) { .\5 }
count of 1 is 3
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

3. WAP to reverse the first m elements of a linked list of n nodes.

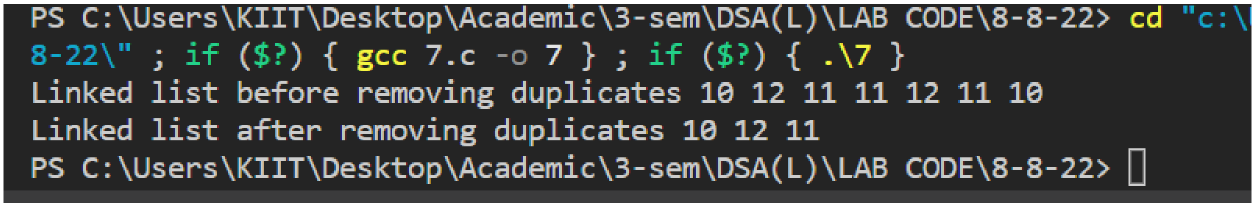
```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int num;
    struct node *next;
};

void create(struct node **);
void reverse(struct node **, int);
void release(struct node **);
void display(struct node *);

int main()
{
    struct node *p = NULL;
    int n;

    printf("Enter data into the list\n");
    create(&p);
    printf("Displaying the nodes in the list:\n");
    display(p);
    printf("Enter the number N to reverse first N node: ");
    scanf("%d", &n);
    printf("Reversing the list...\n");
    if (n > 1)
    {
        reverse(&p, n - 2);
    }
    printf("Displaying the reversed list:\n");
    display(p);
    release(&p);

    return 0;
}

void reverse(struct node **head, int n)
{
    struct node *p, *q, *r, *rear;

    p = q = r = *head;
    if (n == 0)
    {
        q = q->next;
        p->next = q->next;
        q->next = p;
        *head = q;
    }
    else
    {
        p = p->next->next;
        q = q->next;
        r->next = NULL;
        rear = r;
        q->next = r;

        while (n > 0 && p != NULL)
        {
            r = q;
            q = p;
            p = p->next;
            q->next = r;
            n--;
        }
        *head = q;
        rear->next = p;
    }
}

void create(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;

    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}

void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\t", p->num);
        p = p->next;
    }
    printf("\n");
}

void release(struct node **head)
{
    struct node *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
}
```



```
Displaying the nodes in the list:
1       2       3       4       5
Enter the number N to reverse first N node: 3
Reversing the list...
Displaying the reversed list:
3       2       1       4       5
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

4. WAP to remove duplicates from a linked list of n nodes.

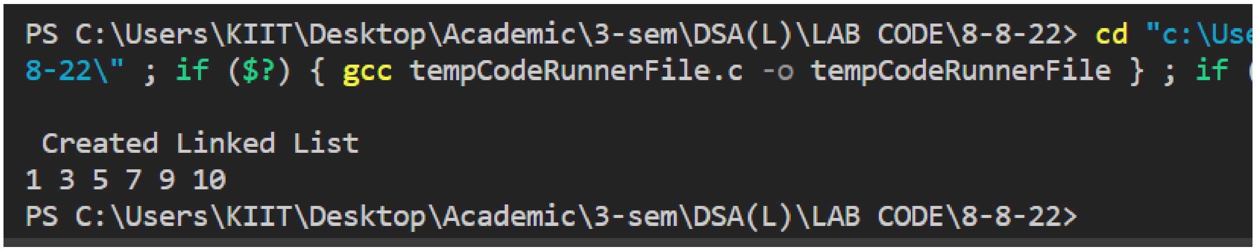
```
#include <stdio.h>
#include <stdlib.h>

/* A linked list node */
typedef struct Node
{
    int data;
    struct Node *next;
} Node;

// Utility function to create a new Node
Node *newNode(int data)
{
    Node *temp = (Node *)malloc(sizeof(Node));
    temp->data = data;
    temp->next = NULL;
    return temp;
}

/* Function to remove duplicates from a
unsorted linked list */
void removeDuplicates(Node *start)
{
    Node *ptr1, *ptr2, *dup;
    ptr1 = start;
    /* Pick elements one by one */
    while (ptr1 != NULL && ptr1->next != NULL)
    {
        ptr2 = ptr1;

        /* Compare the picked element with rest
        of the elements */
        while (ptr2->next != NULL)
        {
            /* If duplicate then delete it */
            if (ptr1->data == ptr2->next->data)
            {
                /* sequence of steps is important here */
                dup = ptr2->next;
                ptr2->next = ptr2->next->next;
                free(dup);
            }
            else /* This is tricky */
                ptr2 = ptr2->next;
        }
        ptr1 = ptr1->next;
    }
}

/* Function to print nodes in a given linked list */
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Driver program to test above function */
int main()
{
    /* The constructed linked list is:
    10->12->11->11->12->11->10*/
    struct Node *start = newNode(10);
    start->next = newNode(12);
    start->next->next = newNode(11);
    start->next->next->next = newNode(11);
    start->next->next->next->next = newNode(12);
    start->next->next->next->next->next = newNode(11);
    start->next->next->next->next->next->next = newNode(10);

    printf("Linked list before removing duplicates ");
    printList(start);

    removeDuplicates(start);

    printf("\nLinked list after removing duplicates ");
    printList(start);

    return 0;
}
```



```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "c:\
8-22\" ; if ($?) { gcc 7.c -o 7 } ; if ($?) { .\7 }
Linked list before removing duplicates 10 12 11 11 12 11 10
Linked list after removing duplicates 10 12 11
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> 
```

5. Given a linked list which is sorted, WAP to insert an element into the linked list in sorted way.

```
#include <stdio.h>
#include <stdlib.h>

/* Link list node */
struct Node
{
    int data;
    struct Node *next;
};


void sortedInsert(struct Node **head_ref,
                  struct Node *new_node)
{
    struct Node *current;
    /* Special case for the head end */
    if (*head_ref == NULL || (*head_ref)->data >= new_node->data)
    {
        new_node->next = *head_ref;
        *head_ref = new_node;
    }
    else
    {
        /* Locate the node before
the point of insertion */
        current = *head_ref;
        while (current->next != NULL && current->next->data < new_node->data)
        {
            current = current->next;
        }
        new_node->next = current->next;
        current->next = new_node;
    }
}

/* BELOW FUNCTIONS ARE JUST UTILITY TO TEST sortedInsert */

/* A utility function to create a new node */
struct Node *newNode(int new_data)
{
    /* allocate node */
    struct Node *new_node = (struct Node *)malloc(
        sizeof(struct Node));

    /* put in the data */
    new_node->data = new_data;
    new_node->next = NULL;

    return new_node;
}

/* Function to print linked list */
void printList(struct Node *head)
{
    struct Node *temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

/* Driver program to test count function*/
int main()
{
    /* Start with the empty list */
    struct Node *head = NULL;
    struct Node *new_node = newNode(5);
    sortedInsert(&head, new_node);
    new_node = newNode(10);
    sortedInsert(&head, new_node);
    new_node = newNode(7);
    sortedInsert(&head, new_node);
    new_node = newNode(3);
    sortedInsert(&head, new_node);
    new_node = newNode(1);
    sortedInsert(&head, new_node);
    new_node = newNode(9);
    sortedInsert(&head, new_node);
    printf("\n Created Linked List\n");
    printList(head);

    return 0;
}
```



```
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22> cd "c:\Use
8-22\" ; if ($?) { gcc tempCodeRunnerFile.c -o tempCodeRunnerFile } ; if (

 Created Linked List
1 3 5 7 9 10
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

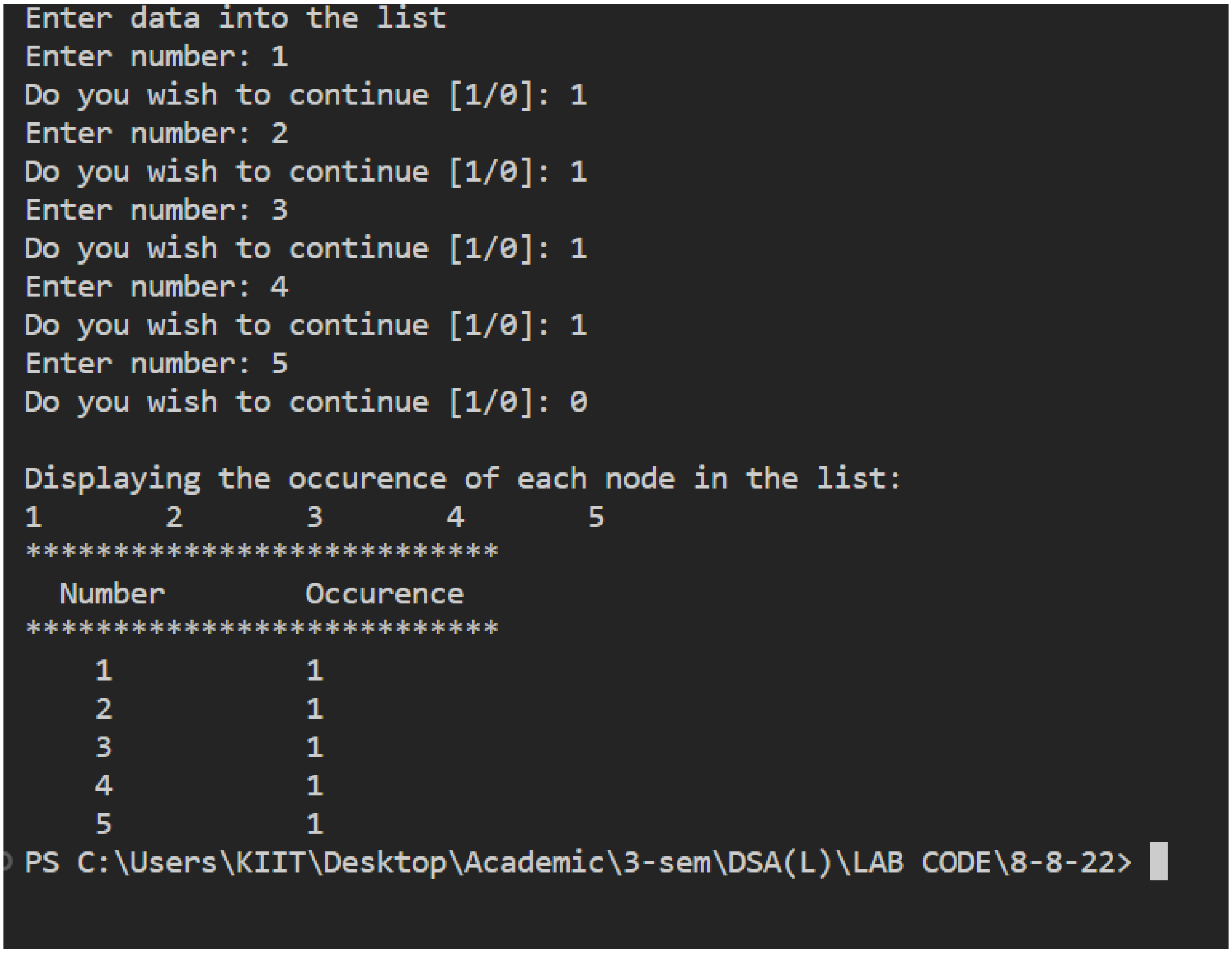
6. WAP to find number of occurrences of all elements in a linked list.

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int num;
    struct node *next;
};

struct node_occur
{
    int num;
    int times;
    struct node_occur *next;
};
```

```
void create(struct node **);
void occur(struct node *, struct node_occur **);
void release(struct node **);
void release_2(struct node_occur **);
void display(struct node *);
void disp_occur(struct node_occur *);

int main()
{
    struct node *p = NULL;
    struct node_occur *head = NULL;
    int n;

    printf("Enter data into the list\n");
    create(&p);
    printf("Displaying the occurence of each node in the list:\n");
    display(p);
    occur(p, &head);
    disp_occur(head);
    release(&p);
    release_2(&head);

    return 0;
}

void occur(struct node *head, struct node_occur **result)
{
    struct node *p;
    struct node_occur *temp, *prev;

    p = head;
    while (p != NULL)
    {
        temp = *result;
        while (temp != NULL && temp->num != p->num)
        {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL)
        {
            temp = (struct node_occur *)malloc(sizeof(struct node_occur));
            temp->num = p->num;
            temp->times = 1;
            temp->next = NULL;
            if (*result != NULL)
            {
                prev->next = temp;
            }
            else
            {
                *result = temp;
            }
        }
        else
        {
            temp->times += 1;
        }
        p = p->next;
    }
}

void create(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;

    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
```

```
}

void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\t", p->num);
        p = p->next;
    }
    printf("\n");
}

void disp_occur(struct node_occur *p)
{
    printf("***************************\n  Number\tOccurence\n***************************\n");
    while (p != NULL)
    {
        printf("    %d\t\t%d\n", p->num, p->times);
        p = p->next;
    }
}

void release(struct node **head)
{
    struct node *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
}

void release_2(struct node_occur **head)
{
    struct node_occur *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
```

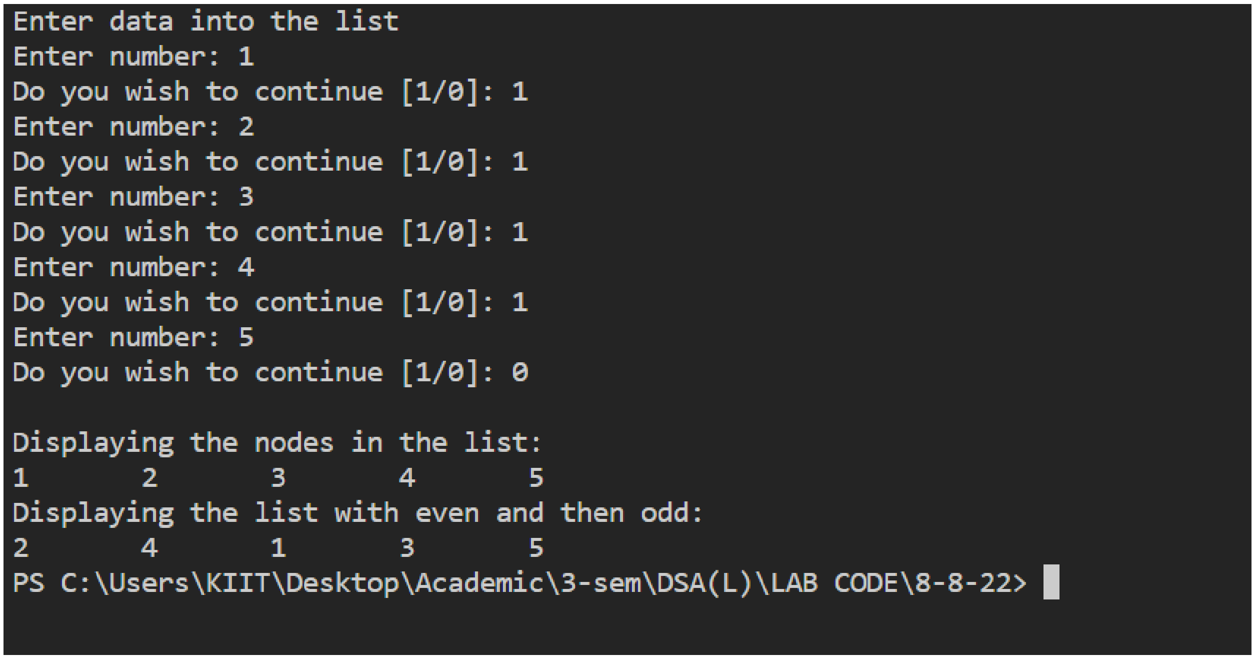
```
}
```



```
Enter data into the list
Enter number: 1
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
Enter number: 5
Do you wish to continue [1/0]: 0

Displaying the occurence of each node in the list:
1       2       3       4       5
***************************
  Number        Occurence
***************************
    1           1
    2           1
    3           1
    4           1
    5           1
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

7. WAP to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list.

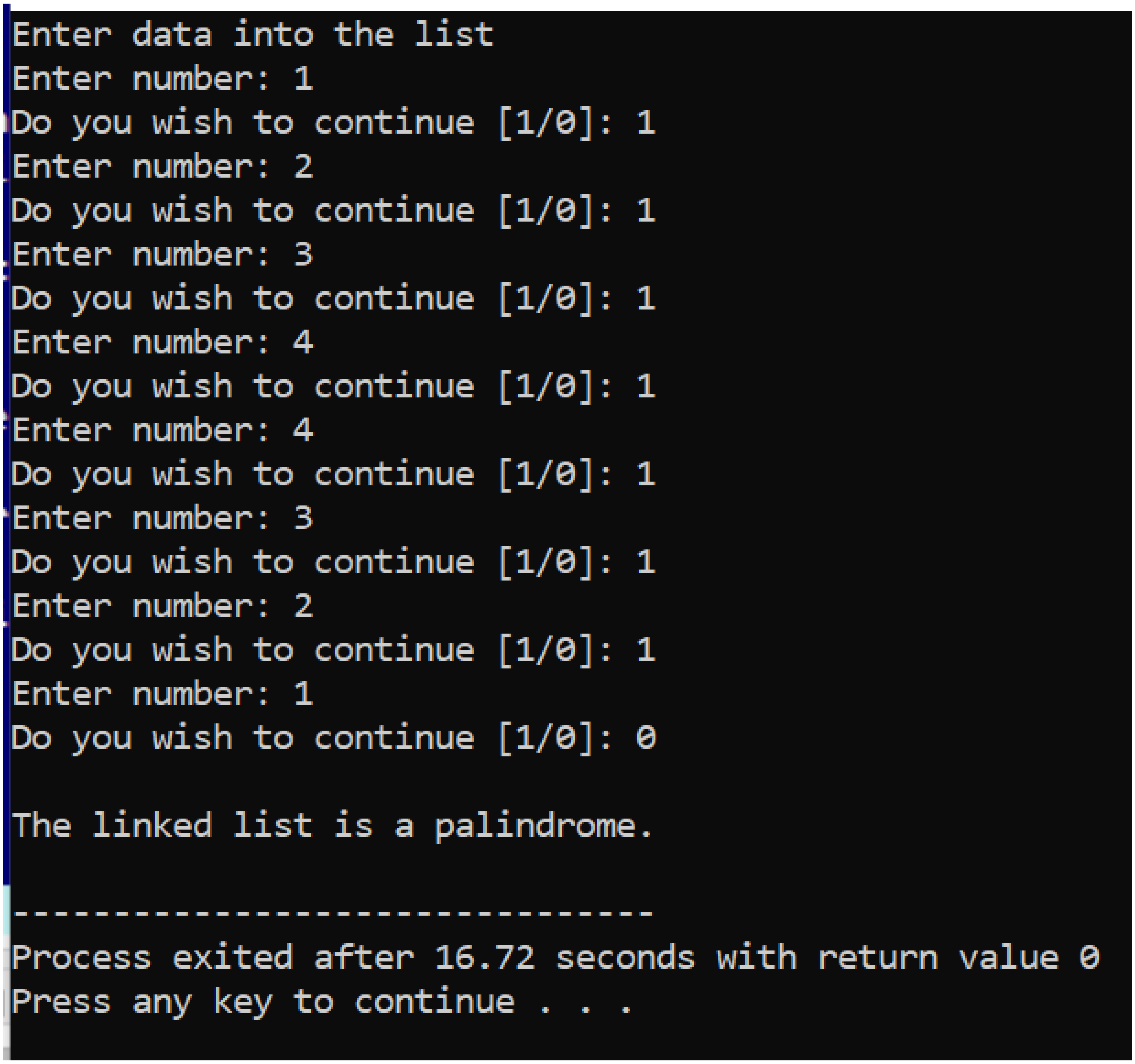
```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int num;
   struct node *next;
};

void create(struct node **);
void generate_evenodd(struct node *, struct node **);
void release(struct node **);
void display(struct node *);

int main()
{
   struct node *p = NULL, *q = NULL;
   int key, result;

   printf("Enter data into the list\n");
   create(&p);
   printf("Displaying the nodes in the list:\n");
   display(p);
   generate_evenodd(p, &q);
   printf("Displaying the list with even and then odd:\n");
   display(q);
   release(&p);

   return 0;
}

void generate_evenodd(struct node *list, struct node **head)
{
   struct node *even = NULL, *odd = NULL, *temp;
   struct node *reven, *rodd;
   while (list != NULL)
   {
       temp = (struct node *)malloc(sizeof(struct node));
       temp->num = list->num;
       temp->next = NULL;
       if (list->num % 2 == 0)
       {
           if (even == NULL)
           {
               even = temp;
           }
           else
           {
               reven->next = temp;
           }
           reven = temp;
       }
       else
       {
           if (odd == NULL)
           {
               odd = temp;
           }
           else
           {
               rodd->next = temp;
           }
           rodd = temp;
       }
       list = list->next;
   }
   reven->next = odd;
   *head = even;
}

void create(struct node **head)
{
   int c, ch;
   struct node *temp, *rear;

   do
   {
       printf("Enter number: ");
       scanf("%d", &c);
       temp = (struct node *)malloc(sizeof(struct node));
       temp->num = c;
       temp->next = NULL;
       if (*head == NULL)
       {
           *head = temp;
       }
       else
       {
           rear->next = temp;
       }
       rear = temp;
       printf("Do you wish to continue [1/0]: ");
       scanf("%d", &ch);
   } while (ch != 0);
   printf("\n");
}

void display(struct node *p)
{
   while (p != NULL)
   {
       printf("%d\t", p->num);
       p = p->next;
   }
   printf("\n");
}

void release(struct node **head)
{
   struct node *temp = *head;
   *head = (*head)->next;
   while ((*head) != NULL)
   {
       free(temp);
       temp = *head;
       (*head) = (*head)->next;
   }
}
```



```
Enter data into the list
Enter number: 1
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
Enter number: 5
Do you wish to continue [1/0]: 0

Displaying the nodes in the list:
1       2       3       4       5
Displaying the list with even and then odd:
2       4       1       3       5
PS C:\Users\KIIT\Desktop\Academic\3-sem\DSA(L)\LAB CODE\8-8-22>
```

```
#include <stdio.h>

#include <stdlib.h>



struct node
```

```
{
    int num;
    struct node *next;
};

int create(struct node **);
int palin_check (struct node *, int);
void release(struct node **);

int main()
{
    struct node *p = NULL;
    int result, count;

    printf("Enter data into the list\n");
    count = create(&p);
    result = palin_check(p, count);
    if (result == 1)
    {
        printf("The linked list is a palindrome.\n");
    }
    else
    {
        printf("The linked list is not a palindrome.\n");
    }
    release (&p);

    return 0;
}

int palin_check (struct node *p, int count)
{
    int i = 0, j;
    struct node *front, *rear;

    while (i != count / 2)
    {
        front = rear = p;
        for (j = 0; j < i; j++)
        {
            front = front->next;
        }
        for (j = 0; j < count - (i + 1); j++)
        {
            rear = rear->next;
        }
        if (front->num != rear->num)
        {
            return 0;
        }
        else
        {
            i++;
        }
    }

    return 1;
}

int create (struct node **head)
{
    int c, ch, count = 0;
    struct node *temp;

    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        count++;
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = *head;
        *head = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    }while (ch != 0);
    printf("\n");

    return count;
}

void release (struct node **head)
{
    struct node *temp = *head;

    while ((*head) != NULL)
    {
        (*head) = (*head)->next;
        free(temp);
        temp = *head;
    }
}
```



```
Enter data into the list
Enter number: 1
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 1
Do you wish to continue [1/0]: 0

The linked list is a palindrome.

--------------------------------
Process exited after 16.72 seconds with return value 0
Press any key to continue . . .
```